Gallager Codes — Recent Results

David J. C. MacKay (mackay@mrao.cam.ac.uk)
Cavendish Laboratory, Cambridge, CB3 0HE, United Kingdom.

Abstract

In 1948, Claude Shannon posed and solved one of the fundamental problems of information theory. The question was whether it is possible to communicate reliably over noisy channels, and, if so, at what rate. He defined a theoretical limit, now known as the Shannon limit, up to which communication is possible, and beyond which communication is not possible. Since 1948, coding theorists have attempted to design error-correcting codes capable of getting close to the Shannon limit.

In the last decade remarkable progress has been made using codes that are defined in terms of sparse random graphs, and which are decoded by a simple probability based message passing algorithm.

This paper reviews low–density parity–check codes (Gallager codes), repeat accumulate codes, and turbo codes, emphasising recent advances. Some previously unpublished results are then presented, describing (a) experiments on Gallager codes with small blocklengths; (b) a stopping rule for decoding of repeat–accumulate codes, which saves computer time and allows block decoding errors to be detected and flagged; and (c) the empirical power laws obeyed by decoding times of sparse graph codes.

1 Introduction

The central problem of communication theory is to construct an encoding and a decoding system that make it possible to communicate reliably over a noisy channel. The encoding system uses the source data to select a codeword from a set of codewords. The decoding algorithm ideally infers, given the output of the channel, which codeword in the code is the most likely to have been transmitted; for an appropriate definition of distance, this is the 'closest' codeword to the received signal. A good code is one in which the codewords are well spaced apart, so that codewords are unlikely to be confused.

Designing a good and practical error correcting code is difficult because (a) it is hard to find an explicit set of well–spaced codewords; and (b) for a generic code, decoding, *i.e.*, finding the closest codeword to a received signal, is intractable.

However, a simple method for designing codes, first pioneered by Gallager (1962), has recently been rediscovered and generalized. The codes are de-

DOCKET
ALARM
Find authenticated court documents without watermarks at docketalarm.com.

fined in terms of sparse random graphs. Because the graphs are constructed randomly, the codes are likely to have well spaced codewords. And because the codes' constraints are defined by a sparse graph, the decoding problem can be solved — almost optimally — by message–passing on the graph. The practical performance of Gallager's codes and their modern cousins is vastly better than the performance of the codes with which textbooks have been filled in the intervening years.

2 Sparse Graph Codes

In a **sparse graph code**, the nodes in the graph represent the transmitted bits and the constraints they satisfy. For a linear code with a codeword length N and rate $R = K/N$, the number of constraints is of order $M = N - K$. [There could be more constraints, if they happen to be redundant.] Any linear code can be described by a graph, but what makes a sparse graph code special is that each constraint involves only a small number of variables in the graph: the number of edges in the graph scales roughly linearly with N, rather than as N^2.

The graph defining a **low–density parity–check code**, or Gallager code (Gallager 1962; Gallager 1963; MacKay 1999), contains two types of node: codeword bits, and parity constraints. In a regular (j, k) Gallager code (figure 1a), each codeword bit is connected to j parity constraints and each constraint is connected to k bits. The connections in the graph are made at random.

Repeat–accumulate codes (Divsalar *et al.* 1998) can be represented by a graph with four types of node (figure 1b): equality constraints $\boxed{=}$, intermediate binary variables (black circles), parity constraints $\boxed{+}$, and the transmitted bits (white circles). The encoder sets each group of intermediate bits to values read from the source. These bits are put through a fixed random permutation. The transmitted stream is the accumulated sum (modulo 2) of the permuted intermediate bits.

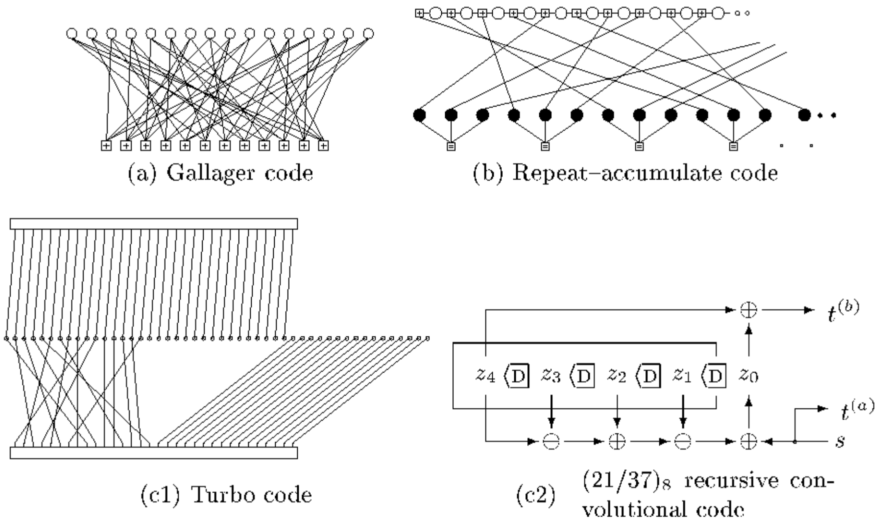
In a **turbo code** (Berrou and Glavieux 1996), the K source bits drive two linear feedback shift registers, which emit parity bits (figure 1c).

All these codes can be decoded by a local message–passing algorithm on the graph, the sum–product algorithm (MacKay and Neal 1996; McEliece *et al.* 1998), and, while this algorithm is not the optimal decoder, the empirical results are record breaking. Figure 2 shows the performance of various sparse graph codes on a Gaussian channel. In figure 2(a) turbo codes with rate 1/4 are compared with regular and irregular Gallager codes over GF(2), GF(8) and GF(16). In figure 2(b) the performance of repeat–accumulate codes of various blocklengths and rate 1/3 is shown.

The best sparse graph codes

Which of the three types of sparse graph code is 'best' depends on the chosen rate and blocklength, the permitted encoding and decoding complexity, and

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.



(a) Gallager code

(b) Repeat–accumulate code

(c1) Turbo code

(c2) $(21/37)_8$ recursive convolutional code

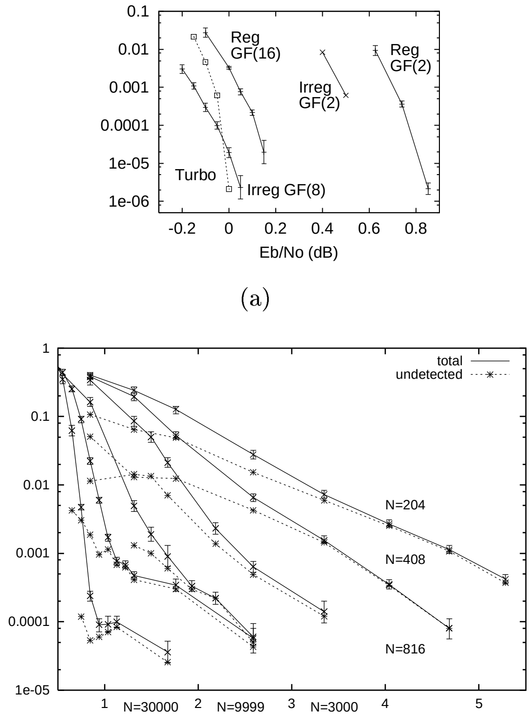
Figure 1. Graphs of three sparse graph codes.

(a) A rate 1/4 low density parity check code (Gallager code) with blocklength $N = 16$, and $M = 12$ constraints. Each white circle represents a transmitted bit. Each bit participates in $j = 3$ constraints, represented by $\boxed{+}$ squares. Each $\boxed{+}$ constraint forces the sum of the $k = 4$ bits to which it is connected to be even.

(b) A repeat–accumulate code with rate 1/3. Each white circle represents a transmitted bit. Each black circle represents an intermediate binary variable. Each $\boxed{=}$ constraint forces the variables to which it is connected to be equal.

(c) A turbo code with rate 1/3. (c1) The circles represent the codeword bits. The two rectangles represent rate 1/2 convolutional codes (c2), with the systematic bits $\{t^{(a)}\}$ occupying the left half of the rectangle and the parity bits $\{t^{(b)}\}$ occupying the right half.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.



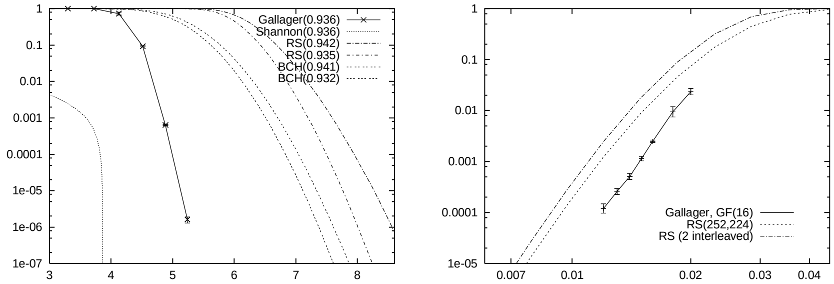
(a)

(b)

Figure 2. (a) Bit error probabilities for communication over a Gaussian channel at rate 1/4: left–right : Irregular LDPC, $GF(8)$, transmitted blocklength 24000 bits; JPL turbo, $N = 65536$ bits (dotted line); Regular LDPC, $GF(16)$, $N = 24448$ bits; Irregular LDPC, $GF(2)$, $N = 64000$ bits; Regular LDPC, $GF(2)$, $N = 40000$ bits. [From Davey and MacKay (1998).]

(b) Block error probability of repeat–accumulate codes with rate 1/3 and various blocklengths, versus E_b/N_0. The dotted lines show the frequency of undetected errors.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.



(a) (b)

Figure 3. (a) A regular binary Gallager code with column weight $j = 4$, rate $R = 0.936$ and blocklength $N = 4376$ ($K = 4094$), compared with BCH codes and interleaved Reed–Solomon codes with similar rates, on a Gaussian channel. Hard–input bounded–distance decoding is assumed for the BCH and RS codes. Vertical axis: block error probability. Horizontal axis: E_b/N_0 [Curves that are further to the left are best.] (b) A Gallager code over GF(16), rate 8/9, blocklength $N = 3996$ bits, applied to a 16–ary symmetric channel, and compared with interleaved RS codes with similar rates. Vertical axis: block error probability. Horizontal axis: channel symbol error probability. [Curves that are further to the right are best.] From MacKay and Davey (1998).

the question of whether occasional undetected errors are acceptable (turbo codes and repeat–accumulate codes both typically make occasional undetected errors, even at high signal–to–noise ratios, because they have a small number of low weight codewords; Gallager codes do not typically show such an error floor).

Gallager codes are the most versatile; it's easy to make a competitive Gallager code with almost any rate and blocklength, as is illustrated in figure 3. Figure 3(a) shows the performance of a high–rate regular binary Gallager code; it outperforms BCH codes and Reed–Solomon codes on this channel. And figure 3(b) shows the performance of a high rate Gallager code over GF(16) on a 16–ary symmetric channel: even though this channel is the sort of channel for which Reed–Solomon codes are intended, the Gallager code still manages to perform a little better than the RS code.

The best binary Gallager codes found so far are *irregular* codes whose parity check matrices have nonuniform weight per column (Luby *et al.* 1998; Urbanke *et al.* 1999). The carefully constructed codes of Urbanke, Richardson and Shokrollahi outperform turbo codes at blocklengths longer than 10^4 bits, with especially impressive results at 10^6 bits, where a rate 1/2 irregular Gallager code has a bit error probability of 10^{-6} at just 0.13 dB from capacity, beating comparable turbo codes by more than 0.3 dB. Turbo codes can

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.